In the Claims:

Please cancel claims 10 and 54-57, without prejudice, amend claims 1-2, 11-12, 21-22, 26-27, 29-30, 32, 35-38, and 43, and add new claim 58. The status of all claims is as follows:

1. (Currently Amended) A method of analyzing a computer program, the computer program including instructions of code, the method comprising:

selecting a number of continuous instructions that are run in program execution order to define an interval of execution;

running the code of the computer program, wherein said running the code comprises running a plurality of continuous instructions of the code in program execution order, wherein within the plurality of continuous instructions, the code is run ~~of the computer program~~ over a plurality of the defined intervals of execution, wherein each of the plurality of defined intervals of execution is run over the selected number of continuous instructions ~~is defined by continuous instructions of the code in program execution order run over a defined amount of time, wherein the amount of time is defined by at least one of a predefined time interval, a predefined number of instructions that are to be run, and a time having a length based on a predefined metric~~;

during said step of running the code, tracking a statistic for a program component;

identifying a behavior of the computer program over each of the plurality of defined intervals of execution over which the code is run based on the tracked statistic;

comparing at least one identified behavior for at least one defined interval of execution over which the code is run to another defined interval of execution over which the code is run to determine similarity between the defined intervals of execution.

2. (Currently Amended) The method of claim 1 wherein said step of running the code comprises at least one of executing the program on hardware, simulating the program's execution in software, direct execution, emulating the program's execution in software, and modeling a hypothetical execution in software.

3. (Original) The method of claim 2 wherein the statistic comprises at least one of a hardware metric and a hardware-independent metric.

4. (Original) The method of claim 3 wherein the statistic comprises at least one of frequency of the component occurring in execution, number of instructions executed, amount of memory used by the program component, time, IPC, performance counters, program counters, and cache miss rate.

5. (Original) The method of claim 2 wherein the program component comprises an identifiable section of control flow of the computer program.

6. (Original) The method of claim 5 wherein the program component comprises at least one of an instruction, basic block, procedure, loop, load instruction, and branch instruction.

7. (Original) The method of claim 2 wherein the program component comprises a memory region.

8. (Original) The method of claim 5 wherein the program component comprises a basic block, the basic block being a section of the code having a single entry point and a single exit.

9. (Previously Presented) The method of claim 2 wherein each of the plurality of intervals of execution is selected independently of particular lines of code of the computer program.

10. (Cancelled)

11. (Currently Amended) The method of claim 1 wherein the defined intervals of execution comprise at least one of overlapping and non-overlapping intervals;

wherein each of the defined intervals of execution comprises a plural number of continuous instructions in program execution order.

12. (Currently Amended) The method of claim 2 further comprising:

based on said comparing step, classifying the plurality of defined intervals of execution over which the code is run into at least one cluster, said one or more clusters being based on similarity of behavior.

13. (Original) The method of claim 12 further comprising:

selecting at least one representative interval of execution for each of the at least one cluster.

14. (Original) The method of claim 13 wherein each of the at least one representative interval of execution is closest to an average behavior of the cluster.

15. (Original) The method of claim 13 wherein the representative interval of execution is the earliest interval of execution within a predetermined distance from an average behavior of the cluster.

16. (Original) The method of claim 13 further comprising:

weighing each of the selected representative intervals of execution based on at least one of a total amount of time, a number of instructions within the cluster, the program component, and the statistic.

17. (Original) The method of claim 16 wherein the weighted representative intervals collectively represent a complete execution of at least a subset of the computer program.

18. (Original) The method of claim 12 further comprising:

minimizing the number of clusters.

19. (Original) The method of claim 17 further comprising:

minimizing the number of clusters.

20. (Original) The method of claim 14 further comprising:

weighing each of the selected representative intervals of execution based on at least one of a total amount of time, a number of instructions within the cluster, the program component, and the statistic;

wherein the weighted representative intervals collectively represent a complete execution of at least a subset of the computer program;

and further comprising minimizing the number of clusters.

21. (Currently Amended) The method of claim 15 further comprising:

weighing each of the selected representative intervals of execution based on at least one of a total amount of time, a number of instructions within the cluster, the program component, and the statistic;

wherein the weighted representative intervals collectively represent a complete execution of at least a subset of the computer program; and further comprising

~~and further comprising~~ minimizing the number of clusters.

22. (Currently Amended) The method of claim ~~10~~ 21 wherein the step of comparing comprises:

comparing each defined interval over which the code is run to the intervals of execution representing at least a subset of execution of the computer program.

23. (Original) The method of claim 22 further comprising:

based on said comparing step, identifying an end of an initialization of the computer program.

24. (Original) The method of claim 22 further comprising:

based on said comparing step, identifying a length of at least one repeating interval of execution.

25. (Original) The method of claim 24 wherein said step of identifying a length comprises performing an analysis of a signal, the signal comprising

differences between each identified interval of execution and the interval of execution representing the at least a subset of execution of the computer program.

26. (Currently Amended) The method of claim 12 further comprising:

determining a confidence and variance by sampling the defined intervals of execution over which the code is run within a particular cluster for at least one of a hardware metric and a hardware-independent metric.

27. (Currently Amended) A method of analyzing a computer program, the computer program including instructions of code, the method comprising:

selecting a number of continuous instructions that are run in program execution order to define an interval of execution;

running the code of the computer program, wherein said running the code comprises running a plurality of continuous instructions of the code in program execution order, wherein within the plurality of continuous instructions, the code is run over a plurality of the defined intervals of execution, wherein each of the plurality of defined intervals of execution is run over the selected number of continuous instructions;

during said step of running the code, tracking a statistic for a program component;

identifying a behavior of the computer program over each of the plurality of defined intervals of execution over which the code is run based on the tracked statistic;

~~A method of analyzing a computer program, the method comprising:~~

~~running at least a portion of the computer program;~~

~~identifying behavior of a hardware-independent metric within at least one arbitrary section of execution of the portion of the computer program during said running step, wherein the at least one arbitrary section of execution is selected independently of particular lines of code of the computer program, wherein each of the arbitrary sections of execution is defined by continuous instructions of the code in program execution order run over a defined amount of time, wherein the amount of time is defined by at least one of a predefined time interval, a predefined number of instructions that are to be run, and a time having a length based on a predefined metric;~~

classifying each of the at least one ~~arbitrary section~~ intervals of execution over which the code is run according to the identified behavior into clusters of similar behavior.

28. (Original) The method of claim 27 wherein said step of identifying comprises identifying a frequency of execution of basic blocks of the executed code, wherein each of the at least one basic block comprises a piece of code of the computer program executed from start to finish, said basic block having only one entry point and one exit.

29. (Currently Amended) The method of claim 28 wherein said step of identifying a frequency provides a group of frequencies for each of the number of defined intervals of execution over which the code is run.

30. (Currently Amended) The method of claim 29 wherein said step of classifying further comprises:

comparing the identified behavior of one of the defined intervals of execution over which the code is run to the identified behavior of another of the defined intervals of execution over which the code is run to identify a phase ~~of the interval~~.

31. (Previously Presented) The method of claim 30 further comprising:

identifying an initialization phase;

determining at least one analysis point occurring after execution of the identified initialization phase.

32. (Currently Amended) The method of claim 28 wherein said step of identifying a frequency of execution for each of the at least one basic block comprises:

for each of the number of defined intervals over which the code is run, determining a interval vector, the interval vector comprising a plurality of ordered

elements, each of the plurality of ordered elements relating to a particular basic block and representing a frequency of execution of the particular basic block.

33. (Original) The method of claim 28 further comprising:

partitioning the computer program into a set of clusters by comparing the determined interval vectors to one another.

34. (Original) The method of claim 33 wherein said step of partitioning further comprises:

determining a group of clusters;

comparing each of the interval vectors to each of the set of clusters;

adding the compared interval vector to a particular cluster based on a goodness of fit between the compared basic block vectors and each of the group of clusters;

changing a centroid of each of the group of clusters;

repeating the comparing, adding, and clustering steps to form the set of clusters.

35. (Currently Amended) A method of analyzing a computer program, the computer program including instructions of code, the method comprising:

selecting a number of continuous instructions that are run in program execution order to define an interval of execution;

running the code, wherein said running the code comprises running a plurality of continuous instructions of the code in program execution order, wherein within the plurality of continuous instructions, the code is run over a plurality of the defined intervals of execution, wherein each of the plurality of defined intervals of execution is run over the selected number of continuous instructions;

during said step of running the code, tracking a statistic for a program component;

identifying a behavior of a hardware-independent metric over each of the plurality of defined intervals of execution over which the code is run based on the tracked statistic;

~~A method of analyzing operation of a computer program, the method comprising:~~

~~executing at least a portion of the computer program;~~

~~for each of a plurality of intervals of execution over the at least a portion of the computer program, identifying behavior of a hardware-independent metric, wherein each of the plurality of intervals of execution is selected independently of particular lines of code within the portion of the computer program, wherein each of the plurality of intervals of execution is defined by continuous instructions of the code in program execution order run over a defined amount of time, wherein the amount of time is defined by at least one of a predefined time interval, a predefined number of instructions that are to be run, and a time having a length based on a predefined metric;~~

identifying behavior of the hardware-independent metric over full execution of the ~~at least a portion of the computer program~~ code to identify a target behavior;

comparing the identified behavior of each of the plurality of defined intervals of execution over which the code is run to the identified target behavior over full execution of the code to determine a representative interval;

simulating execution of the computer program over the determined representative interval.

36. (Currently Amended) The method of claim 35 wherein said step of identifying comprises:

deriving a plurality of basic block vectors, each basic block vector representing code blocks of the program executed during the defined interval of execution over which the code is run, the basic block vector being based on frequencies of basic blocks of executed code within execution of the program;

wherein the basic block vector comprises a single dimensional array where a single element in the array exists for a basic block in the program.

37. (Currently Amended) The method of claim ~~2–~~35wherein the method is performed in run-time.

38. (Currently Amended) The method of claim 37 wherein said step of identifying comprises tracking a proportion of instructions executed from different sections of code of the program over which each of the defined plurality of intervals is run;

further comprising, for each of the defined ~~interval~~plurality of intervals over which the code is run, classifying the identified behavior into phases corresponding to changes in behavior across the executed code.

39. (Original) The method of claim 38 further comprising:

predicting when execution of the code is about to enter a phase change;

predicting a phase entered by the phase change.

40. (Original) The method of claim 38 wherein said step of identifying comprises, for each section of code:

capturing an identifier of the section of code;

capturing a number of instructions executed for the section of code.

41. (Previously Presented) The method of claim 40 further comprising:

reducing a number of the identified sections of code to a lower number.

42. (Previously Presented) The method of claim 40 further comprising:

comparing each section of code to a section of code in a history;

if the compared section of code varies from the section of code in the history by a greater amount than a predetermined threshold, adding the comparedsection of code to the history.

43. (Currently Amended) The method of claim 2 wherein the behavior identified for ~~an~~ a particular one of the defined plurality of intervals of execution over which the code is run ~~interval~~ is collected in a vector, the vector containing the statistic for at least one element representing at least one component.

44. (Original) The method of claim 43 wherein the vector is retained in at least one of a memory, a storage medium, and a table.

45. (Previously Presented) The method of claim 44 wherein the vector is stored as a signature that represents at least one of the behavior of a complete vector, a projection of the vector, a compressed representation of the vector, a partial representation of the vector, and a subset of the identified behavior collected in the vector.

46. (Original) The method of claim 45 further comprising:

storing a phase ID with the signature, wherein the phase ID comprises at least one of a complete signature, a subset of the signature, a partial representation of the signature, a name independent of the signature, and a number.

47. (Previously Presented) The method of claim 46 wherein the stored phase ID is identified for an interval by looking up the signature in storage and based on said looking up, either using the phase ID stored with the signature or creating a new phase ID.

48. (Original) The method of claim 47 wherein the identified behavior and the tracked statistic for at least one interval with a phase ID are stored and associated with one another.

49. (Original) The method of claim 48 wherein, if a storage area for storing the phase ID, behavior, and statistic is finite, only a single stored signature for a phase ID, and the phase ID, are stored.

50. (Original) The method of claim 47 further comprising, after the phase ID is identified by a signature for an interval:

looking up the phase ID to find the associated statistic for the interval.

51. (Original) The method of claim 50 further comprising:

using the found associated statistic, performing at least one of a behavior optimization, statistic optimization, load-time optimization, run-time optimization, and hardware reconfiguration.

52. (Original) The method of claim 47 wherein the phase ID is stored in a prediction table, and further comprising:

predicting a phase ID for an interval using the stored phase ID.

53. (Original) The method of claim 52 further comprising:

retrieving information for the predicted phase ID;

using the retrieved information, guiding optimization for the computer program.

54. (Cancelled)

55. (Cancelled)

56. (Cancelled)

57. (Cancelled)

58. (New) A method of analyzing a computer program, the computer program including instructions of code, the method comprising:

selecting an amount of execution time over which continuous instructions of the code are run in program execution order to define an interval of execution;

running the code of the computer program, wherein said running the code comprises running a plurality of continuous instructions of the code in program execution order, wherein within the plurality of continuous instructions, the code is run over a plurality of the defined intervals of execution, wherein each of the plurality of defined intervals of execution is run over the selected amount of execution time;

during said step of running code, tracking a statistic for a program component;

identifying a behavior of the computer program over each of the defined plurality of intervals of execution over which the code is run based on the tracked statistic;

comparing at least one identified behavior for at least one defined interval of execution over which the code is run to another defined interval of execution over which the code is run to determine similarity between the intervals of execution.